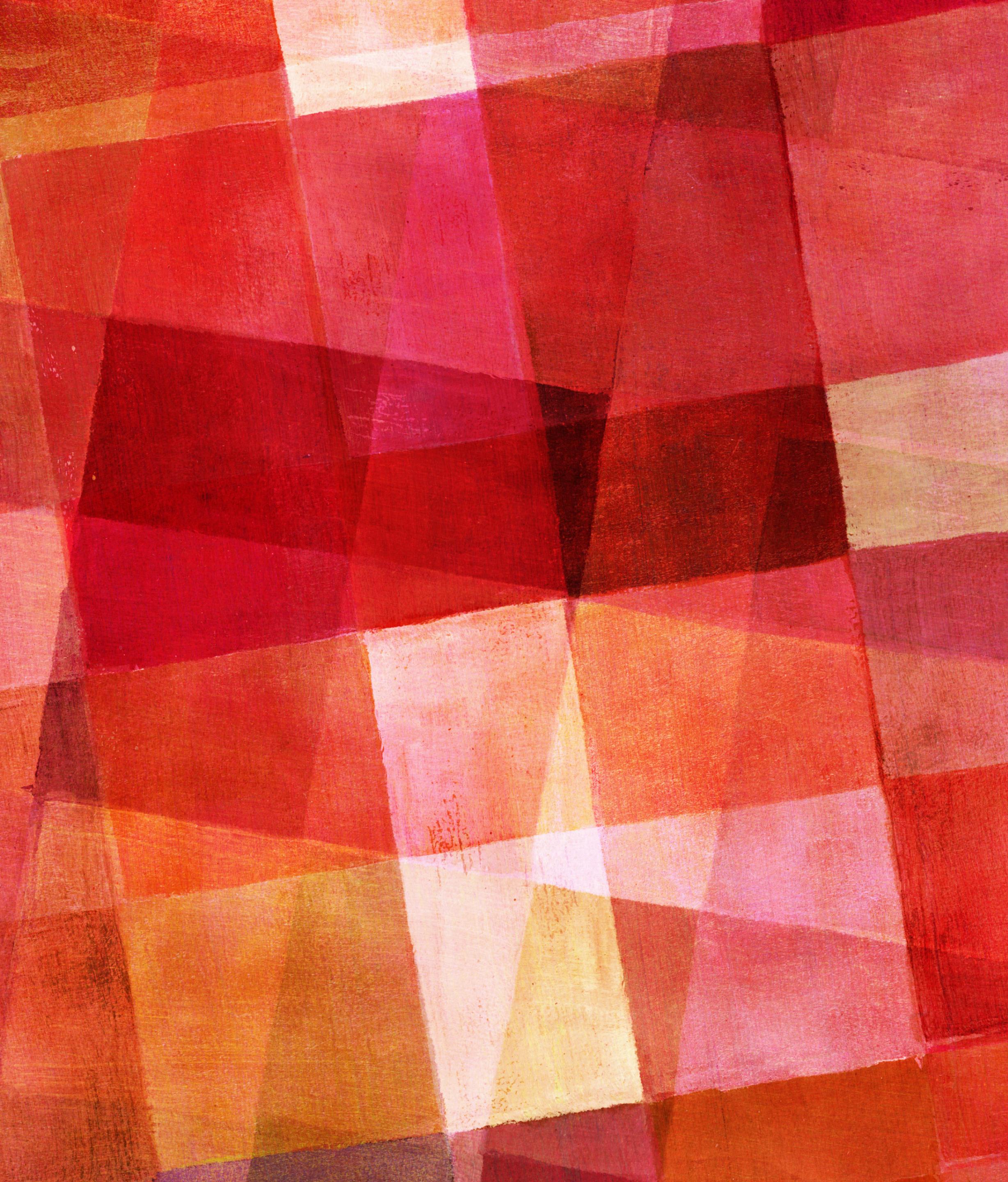




PROGRAMAÇÃO EM LINGUAGEM SQL



ORDER BY

.....

ORDER BY

Classifica os dados retornados por uma consulta no SQL Server.

O uso desta cláusula serve para:

- Ordenar o conjunto de resultados de uma consulta pela lista de colunas específica. A ordem na qual as linhas são retornadas num conjunto de resultados não é garantida, a menos que uma cláusula ORDER BY seja especificada.*
- Determinar a ordem em que os valores da função de classificação são aplicados ao conjunto de resultados.*

ORDER BY

Order by [nomeColuna]

```
SELECT prod_name  
FROM products  
ORDER BY prod_name
```

Order by [nomeColuna1, nome coluna2]

```
SELECT prod_id, prod_price, prod_name  
FROM products  
ORDER BY prod_price, prod_name
```

ORDER BY

Order by [númeroColuna]

```
SELECT prod_name  
FROM products  
ORDER BY 1
```

Order by [númeroColuna2, númeroColuna3]

```
SELECT prod_id, prod_price, prod_name  
FROM products  
ORDER BY 2, 3
```

ORDER BY

Especificando a direção de classificação #

A direção de classificação padrão é crescente, de A a Z para caracteres ou do menor ao maior para números.

*Podemos alterar a direção da classificação usando as palavras-chave **ASC** para **DESC** ascendente para descendente junto com a cláusula ORDER BY.*

ORDER BY

Order by [nomeColuna desc]

```
SELECT prod_id, prod_price,  
prod_name  
FROM products  
ORDER BY prod_price DESC
```

Order by [nomeColuna asc]

```
SELECT prod_id, prod_price,  
prod_name  
FROM products  
ORDER BY prod_price ASC
```

WHERE

A cláusula WHERE é usada para filtrar os dados retornados do banco de dados.

Será necessário especificar as colunas e condições para limitar os dados que serão retornados.

Esta cláusula WHERE oferece suporte a vários tipos diferentes de operadores além do operador de igualdade simples que usamos até agora.

Operator	Description
=	Equality
<>	Not equal
!=	Not equal
<	Less than
>	Greater than
!<	Not less than
!>	Not greater than
>=	Greater than or equal to
<=	Less than or equal to
BETWEEN	Between two specified values
IS NULL	Is a NULL value

WHERE

WHERE [nomecoluna]
(operador) valor

```
SELECT prod_name, prod_price  
FROM products  
WHERE prod_price < 10
```

WHERE [nomecoluna]
(operador) valor

```
SELECT prod_name, prod_price  
FROM products  
WHERE prod_price <= 10
```

WHERE

**WHERE [nomecoluna]
(operador) valor**

SELECT vend_id, prod_name

FROM products

WHERE vend_id <> 'DLL01'

WHERE

**WHERE [nomecoluna]
(operador) valor**

```
SELECT prod_name, prod_price  
FROM products
```

```
WHERE prod_price BETWEEN  
5 AND 10
```

WHERE

WHERE [nomecoluna]
(operador) valor

```
SELECT prod_name  
FROM products  
WHERE prod_price IS NULL
```

WHERE [nomecoluna]
(operador) valor

```
SELECT cust_name  
FROM customers  
WHERE cust_email IS NULL
```

WHERE + (AND/OR)

```
WHERE [nomecoluna]
      (operador) valor AND/OR
      [nomecoluna] (operador) valor
SELECT prod_id, prod_price,
prod_name
FROM products
WHERE vend_id = 'DLL01'
AND prod_price <= 4;
```

```
WHERE [nomecoluna]
      (operador) valor AND/OR
      [nomecoluna] (operador) valor
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 'DLL01' OR
vend_id = 'BRS01'
```

WHERE + (IN)

WHERE [nomecoluna] IN(valor)

SELECT prod_name, prod_price

FROM products

WHERE vend_id **in** ('DLL01',
'BSR01')

WHERE + (NOT)

WHERE NOT [nomecoluna]
operador Valor

```
SELECT prod_name  
FROM products  
WHERE NOT vend_id =  
'DLL01'
```

WHERE NOT [nomecoluna]
operador Valor

```
SELECT prod_name  
FROM products  
WHERE NOT vend_id <>  
'DLL01'
```

WHERE + (LIKE)

**WHERE [nomecoluna] LIKE
'Valor%'** (Valor encontrado no
inicio de algum campo a ser
analisado)

```
SELECT prod_id, prod_name  
FROM products
```

```
WHERE prod_name LIKE  
'Fish%'
```

**WHERE [nomecoluna] LIKE
'%Valor'** (Valor encontrado no
fim de algum campo a ser
analisado)

```
SELECT prod_id, prod_name  
FROM products
```

```
WHERE prod_name LIKE  
'%Fish'
```

WHERE + (LIKE)

WHERE [nomecoluna] LIKE ‘%Valor%’ (Valor encontrado no inicio/meio/fim de algum campo a ser analisado)

SELECT prod_id, prod_name

FROM products

WHERE prod_name **LIKE**
'%bean bag%'

WHERE + (LIKE) - WILDCARD

**WHERE [nomecoluna] LIKE
'[Valor1/Valor2]%' (O que
procuramos poderá começar por
Valor 1 ou por Valor 2)**

SELECT cust_contact

FROM customers

**WHERE cust_contact LIKE
'[JM]%'**

**WHERE [nomecoluna] LIKE
'[Valor1/Valor2]%' (O que
procuramos não poderá começar
por Valor 1 ou por Valor 2)**

SELECT cust_contact

FROM customers

**WHERE cust_contact LIKE
'[^JM]%'**

SELECT + TOP

Usar a clausula TOP permite que sejam retornados todos os registos que foram indicados na clausula TOP.

TOP(<number>)

SELECT + TOP

SELECR TOP (VALOR) *

FROM TABELA

SELECT TOP 5 *

FROM products

SELECT + UPPER AND LOWER

Usar a clausula UPPER e LOWER para alterar uma string para maiúsculas ou minúsculas.

Âmbito: Poderá ser necessário apresentar todos os dados em maiúsculas num relatório, por exemplo.

UPPER(<string>) LOWER(<string>)

SELECT + UPPER AND LOWER

```
USE AdventureWorks2019;
```

```
GO
```

```
SELECT LastName,
```

```
    UPPER(LastName) AS "UPPER",
```

```
    LOWER(LastName) AS "LOWER"
```

```
FROM Person.Person
```

SELECT + LEFT AND RIGHT

As funções LEFT e RIGHT retornam um número específico de caracteres no lado esquerdo ou direito de uma string.

Com estas cláusulas é possível apenas retornar um numero específico de uma determinada string.

LEFT(<string>,<number of characters>) RIGHT(<string>,<number of characters>)

SELECT + LEFT AND RIGHT

USE AdventureWorks2019;

GO

SELECT LastName,**LEFT**(LastName,5) **AS** "LEFT",

RIGHT(LastName,4) **AS** "RIGHT"

FROM Person.Person

WHERE BusinessEntityID **IN**

(293,295,211,297,299,3057,15027);

SELECT + LEN() OU DATALENGTH()

*Usar LEN para retornar o número de caracteres numa string.
DATALENGTH retorna o número de bytes numa string.*

LEN(<string>)DATALENGTH(<string>)

SELECT + LEN() OU DATALENGTH()

USE AdventureWorks2019;

GO

SELECT LastName,**LEN**(LastName) **AS** "Length",

DATALENGTH(LastName) **AS** "Data Length"

FROM Person.Person

WHERE BusinessEntityID **IN** (293,295,211,297,299,3057,15027);

SELECT + CHARINDEX()

Usar CHARINDEX para encontrar a posição numérica inicial de uma string.

CHARINDEX(<search string>, <target string>[, <start location>])

SELECT + CHARINDEX()

USE AdventureWorks2019;

GO

```
SELECT LastName, CHARINDEX('e',LastName) AS "Find e",
CHARINDEX('e',LastName,4) AS "Skip 4 Characters",
CHARINDEX('be',LastName) AS "Find be",
CHARINDEX('Be',LastName) AS "Find B"
FROM Person.Person
WHERE BusinessEntityID IN (293,295,211,297,299,3057,15027);
```

SELECT + SUBSTRING

Usar SUBSTRING para retornar uma parte de uma string começando numa determinada posição e para um número específico de caracteres.

SUBSTRING(<string>,<start location>,<length>)

SELECT + SUBSTRING

USE AdventureWorks2019

GO

```
SELECT LastName, SUBSTRING(LastName,1,4) AS "First 4",  
       SUBSTRING(LastName,5,50) AS "Characters 5 and later"  
FROM Person.Person  
WHERE BusinessEntityID IN (293,295,211,297,299,3057,15027);
```

SELECT + REVERSE

REVERSE retorna uma string na ordem inversa.

SELECT REVERSE('!dlroW ,olleH')

SELECT + REPLACE

Usar REPLACE para substituir um valor de string por outro.

REPLACE tem três parâmetros obrigatórios.

Usar o REPLACE para limpar os dados; por exemplo, pode precisar de substituir barras (/) numa coluna de número de telefone por hifens (-) para um relatório.

REPLACE(<string value>,<string to replace>,<replacement>)

SELECT + REPLACE

```
USE AdventureWorks2012;
```

```
GO
```

```
--1
```

```
SELECT LastName,  
REPLACE(LastName,'A','Z') AS "Replace  
A",  
REPLACE(LastName,'A','ZZ') AS  
"Replace with 2 characters",  
REPLACE(LastName,'ab','') AS  
"Remove string"  
FROM Person.Person
```

```
--2
```

```
SELECT  
BusinessEntityID,LastName,MiddleName,  
REPLACE(LastName,'a',MiddleName)  
AS "Replace with MiddleName",  
REPLACE(LastName,MiddleName,'a')  
AS "Replace MiddleName"  
FROM Person.Person
```

SELECT + GETDATE() / SYSDATETIME()

GETDATE e SYSDATETIME

Usar GETDATE ou SYSDATETIME para retornar a data e hora atuais do servidor.

A diferença é que SYSDATETIME retorna sete casas decimais após o segundo, enquanto GETDATE retorna apenas três casas.

GETDATE e SYSDATETIME são funções não determinísticas. Isso significa que eles retornam valores diferentes cada vez que são chamados.

SELECT GETDATE(), SYSDATETIME();

SELECT + DATEADD()

Usar DATEADD para adicionar várias unidades de tempo a uma data. A função requer três parâmetros: a parte da data, o número e uma data.

T-SQL não tem uma função DATESUBTRACT, mas é possível usar um número negativo para realizar a mesma coisa.

Âmbito: Pode usar DATEADD para calcular uma data expirada ou uma data de vencimento de um pagamento.

DATEADD(<date part>,<number>,<date>)

SELECT + DATEADD()

Use AdventureWorks2019

GO

--1

```
SELECT OrderDate, DATEADD(year,1,OrderDate) AS OneMoreYear,  
       DATEADD(month,1,OrderDate) AS OneMoreMonth,  
       DATEADD(day,-1,OrderDate) AS OneLessDay
```

FROM Sales.SalesOrderHeader

WHERE SalesOrderID in (43659,43714,60621);

--2

```
SELECT DATEADD(month,1,'1/29/2009') AS FebDate;
```

SELECT + DATEDIFF()

A função DATEDIFF permite encontrar a diferença entre duas datas.

A função requer três parâmetros: a parte da data e as duas datas.

Âmbito: A função DATEDIFF pode ser usada para calcular quantos dias se passaram desde que os pedidos não enviados foram recebidos, por exemplo.

DATEDIFF(<datepart>,<early date>,<later date>)

SELECT + DATENAME()/DATEPART()

As funções DATENAME e DATEPART retornam a parte da data especificada.

Âmbito: Usam as funções DATENAME e DATEPART para exibir apenas o ano ou mês nos relatórios, por exemplo.

DATEPART sempre retorna um valor numérico.

DATENAME retorna o nome quando a parte da data é o mês ou o dia da semana.

DATENAME(<datepart>,<date>) DATEPART(<datepart>,<date>)

SELECT + DATENAME()/DATEPART()

Use AdventureWorks2019

GO

--1

```
SELECT OrderDate, DATEPART(year,OrderDate) AS  
OrderYear,  
DATEPART(month,OrderDate) AS OrderMonth,  
DATEPART(day,OrderDate) AS OrderDay,  
DATEPART(weekday,OrderDate) AS OrderWeekDay  
FROM Sales.SalesOrderHeader  
WHERE SalesOrderID in (43659,43714,60621);
```

--2

```
SELECT OrderDate, DATENAME(year,OrderDate)  
AS OrderYear,  
DATENAME(month,OrderDate) AS OrderMonth,  
DATENAME(day,OrderDate) AS OrderDay,  
DATENAME(weekday,OrderDate) AS  
OrderWeekDay  
FROM Sales.SalesOrderHeader  
WHERE SalesOrderID in (43659,43714,60621);
```

SELECT + DAY, MONTH, AND YEAR

As funções DIA, MÊS e ANO funcionam exatamente como DATEPART.

Essas funções são apenas formas alternativas de obter o dia, mês ou ano de uma data.

DAY(<date>) MONTH(<date>) YEAR(<date>)

SELECT + DAY, MONTH, AND YEAR

Use AdventureWorks2019

GO

```
SELECT OrderDate, YEAR(OrderDate) AS OrderYear,  
MONTH(OrderDate) AS OrderMonth,  
DAY(OrderDate) AS OrderDay  
FROM Sales.SalesOrderHeader  
WHERE SalesOrderID in (43659,43714,60621);
```

SELECT + CONVERT

A função CONVERT tem um parâmetro opcional chamado style que pode ser usado para formatar uma data.

Com esta função é possível formar datas juntamente com a função DATEPART, bem como converter a data para um formato mais desejado

CONVERT(<data type, usually varchar>,<date>,<style>)

SELECT + CONVERT

--1 The hard way!

```
SELECT CAST(DATEPART(YYYY,GETDATE()) AS  
VARCHAR) + '/' +  
CAST(DATEPART(MM,GETDATE()) AS VARCHAR) +  
'/' + CAST(DATEPART(DD,GETDATE()) AS  
VARCHAR) AS DateCast;
```

--2 The easy way!

```
SELECT CONVERT(VARCHAR,GETDATE(),111) AS  
DateConvert;
```

--3

```
USE AdventureWorks2019
```

```
GO
```

```
SELECT CONVERT(VARCHAR,OrderDate,1) AS "1",  
CONVERT(VARCHAR,OrderDate,101) AS "101",  
CONVERT(VARCHAR,OrderDate,2) AS "2",  
CONVERT(VARCHAR,OrderDate,102) AS "102"  
FROM Sales.SalesOrderHeader  
WHERE SalesOrderID in (43659,43714,60621);
```

SELECT + FORMAT

O objetivo principal é simplificar a conversão de valores de data / hora em valores de string.

Outro objetivo da função de formato é converter valores de data / hora em equivalências culturais.

FORMAT (value, format [, culture])

USE AdventureWorks2019

GO

```
SELECT FORMAT( GETDATE(),'dd', 'en-US' ) AS Result,  
SELECT FORMAT( GETDATE(), 'd/M/y', 'en-US' ) AS Result,  
SELECT FORMAT( GETDATE(), 'dd/MM/yyyy', 'en-US' ) AS Result  
FROM Sales.SalesOrderHeader  
WHERE SalesOrderID in (43659,43714,60621);
```

SELECT + FORMAT

SELECT DATEFROMPARTS(2012, 3, 10) AS RESULT

SELECT TIMEFROMPARTS(12, 10, 32, 0, 0) AS RESULT

SELECT DATETIME2FROMPARTS (2012, 3, 10, 12, 10, 32, 0, 0) AS RESULT